Jurnal Teknologi, 40(D) Jun. 2004: 25–36

CHAIN CODE ALGORITHM IN DERIVING T-JUNCTION AND REGION OF A FREEHAND SKETCH

HABIBOLLAH HARON[1], SITI MARIYAM SHAMSUDDIN[2] & DZULKIFLI MOHAMED[3]

Abstract. Chain code scheme is one of the Picture Description Languages used to represent lines. This paper focuses on the line drawing interpretation by utilising closed loop chain code algorithm as a tool in deriving two geometric entities, i.e. T-junction and region, of a two-dimensional line drawing sketch. The closed loop chain code algorithm assumes that the sketch represents a three-dimensional solid object. The algorithm is based on 8-connected 3×3 windows of Freeman chain code. Two factors determine the success of the algorithm. The first factor is the direction of traverses either clockwise or anti-clockwise. The other is the start location of the 3×3 window traverse. This paper explains these factors and their applications in deriving the geometric entities. The discussion is supported with an example of L-block object for clarification in the presentation of the algorithm. The paper is concluded with conclusion and future works.

Keywords: Line drawing interpretation, chain code, L-block object, Picture Description Languages

Abstrak. Perwakilan kod rantai ialah salah satu daripada Bahasa Penghuraian Gambar yang digunakan untuk mewakili garisan. Kertas kerja ini memfokuskan kepada terjemahan lukisan garisan dengan menerapkan algoritma kod rantai dalam mendapatkan dua entiti geometri, iaitu simpang-T dan kawasan, bagi lakaran lukisan garisan dua dimensi. Algoritma kod rantai tertutup ini mengandaikan bahawa lakaran mewakili objek padu tiga dimensi. Algoritma ini ialah berasaskan kod rantai Freeman 8-kaitan dalam tetingkap 3×3. Terdapat dua faktor yang menentukan kejayaan algoritma ini. Pertama ialah arah penjelajahan sama ada ikut jam atau lawan jam. Kedua ialah lokasi permulaan bagi penjelajahan dalam tetingkap 3×3 tersebut. Kertas kerja ini menerangkan faktor-faktor ini dan aplikasi mereka dalam mendapatkan entiti geometri tersebut. Perbincangan ini dibantu oleh contoh lakaran objek blok-L untuk memudahkan pemahaman terhadap algoritma. Kertas kerja ini diakhiri dengan kesimpulan dan cadangan pembaikan.

Kata kunci: Terjemahan lukisan garisan, kod rantaian, objek blok-L, Bahasa Penghuraian Gambar

1.0 INTRODUCTION

Computerized line drawing interpretation involves representation of the line using Picture Representation Language (PDL) such as line labeling and chain code scheme. Based on the representation, sketch interpretation such as detection of corner, derivation of depth, validation and classification of the type of line drawing, and recovery of

[1,2&3] Faculty of Computer Science & Information Systems, Universiti Teknologi Malaysia. 81310 UTM Skudai, Johor, Malaysia. E-mail: habib@fsksm.utm.my, mariyam@fsksm.utm.my, dzul@fsksm.utm.my. Phone: 07-5532349. Fax: 07-5565044

hidden parts can be carried out. This paper presents chain code as an input of the line drawing interpretation.

Chain code scheme is a representation that consists of series of numbers. The numbers represent direction of the next pixel that can be used to represent shape and input format for numerous shape analysis algorithms. Since introduced by Freeman [1, 2], works on chain code in representing images, describing and recognizing shapes have been given consideration as an alternative method. Works on new scheme of chain code also have been proposed by Bribiesca [3, 4], Chen [5], Arrebola[6], O'Gorman[7], and Nunes [8], as well as techniques to compress the chain code by Choo [9].

In terms of application of chain code, Bribiesca [3, 10] used chain code to represent three-dimensional curve and surface coverage, Chen [5] used it to represent three-dimensional curve. Choo [11] used chain code to represent polycurve, Livarinen [12], to recognize shape of irregular object, Pitas [13], to describe object, Arrebola [6,14], to detect corner and represent curve, Hanyu [15], to recognize object, and Maeder [16] used chain code in animation technique. They show that most applications of chain code are to represent curve and recognize shape of object because chain code allows reduction of data while preserving the information. Based on these works, we propose an enhanced chain code algorithm called closed loop chain code in representing a curve as an input in the line drawing interpretation and 3D object reconstruction. This paper is based on 8-connected chain code introduced by Freeman [1] in extracting geometric entities, namely, junctions, lines, and regions.

This section explains previous works on chain code and its application in line drawing interpretation. Section 2 gives the definition of chain code. Sections 3 and 4 present the algorithm to extract the geometric entities in detail, supported by example for clarification. Section 5 discusses the results of the algorithm while Section 6 concludes this paper and presents the future works.

Three assumptions have been made on this work. Firstly, the input image has already passed the digitization and thinning process. The details of the process can be referred in Haron [17]. Secondly, the two-dimensional line drawing represents a three-dimensional object. Thirdly, the thinning process has removed all unwanted pixels.

2.0 CHAIN CODE

Chain code is a list of codes ranging from 0 to 7 in clockwise direction. These codes represent direction of the next pixel connected in 3×3 windows as shown in Figure 1. For example, if a current pixel in an image is located at coordinate (5,5), the coordinate of the next pixel based on the chain code is given by Table 1. The coordinate of the next pixel is calculated based on the adding and subtraction of column and row by 1, depending on the value of chain code. This representation is based on work by Freeman chain code [2].

	Column-1	Column	Column+1
row-1	5	6	7
row	4	(current pixel)	0
row+1	3	2	1

Figure 1 Location of chain code

Table 1 Relation of pixel and chain code

Current pixel at coordinate (5,5)		
Code	**Next row**	**Next column**
0	5	6
1	6	6
2	6	5
3	6	4
4	5	4
5	4	4
6	4	5
7	4	6

There are eight locations of pixels that surround the current pixel and they are represented by row and column. The chain code represents the direction of the next pixel. For example, in code 0, the direction of the next pixel is located at coordinate (row, column+1). Code –1 indicates that the current pixel is the last pixel in the list.

3.0 DERIVING T-JUNCTIONS

This section discusses the algorithm to derive T-junctions with an example for better understanding. Figure 2 shows the actual image drawn using Adobe Photoshop in tiff format. It was assumed that the image was already thinned. Figure 3 shows the thinned binary image of Figure 2. The image is already modified by removing the '0' pixel, and adding the column and row header for better understanding. The discussion in Section 3 and 4 is based on the pixel and their locations.

In deriving the T-junctions, series of chain code represents boundary lines and internal lines that are derived in clockwise direction. The steps involved are given in Algorithm 1.

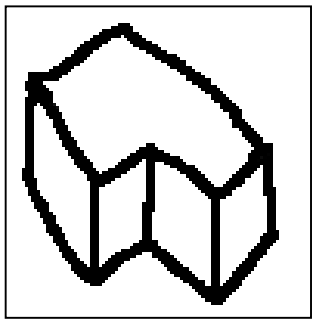

Figure 2 L-block size 100×100

Step 1: Traverse the thinned image. Assume start code is 0.
Step 2: For the first pixel '1' found; assign the location as current pixel. Start the clockwise search to find the next '1' pixel. Table 2 shows the first start test location.
Step 3: Change the pixel to '2' to indicate that the location is already traversed. Put the code of the location in the series of the chain code. Assign this location as location of current pixel.
Step 4: Move the location of current pixel to the next pixel found. Repeat step 2 until traverse back to the start pixel or when the pixel '2' is found.

Algorithm 1 Deriving T-junction

Table 2 Start test location (clockwise direction)

Previous code	0	1	2	3	4	5	6	7
Start test location	5	6	7	0	1	2	3	4

Based on the algorithm shown in Algorithm 1, T-junctions of thinned binary image shown in Figure 3 are searched. The steps taken are listed as follows:

Step 1: Assume start code is 0.
Step 2: The first pixel found is at location (7,38). From location (7,38) there are two pixels connected namely at location (8,37) and (7,39). By applying Table 2, since the previous code is 0, then the start test location is at code 5. From location (6,37), traverse the pixel in clockwise direction.
Step 3: The next pixel found is at location (7,39) and the code is 0. The pixel is changed to number 2 to indicate that the pixel is already traversed. The current location is now changed to (7,39) and the current code is 0.

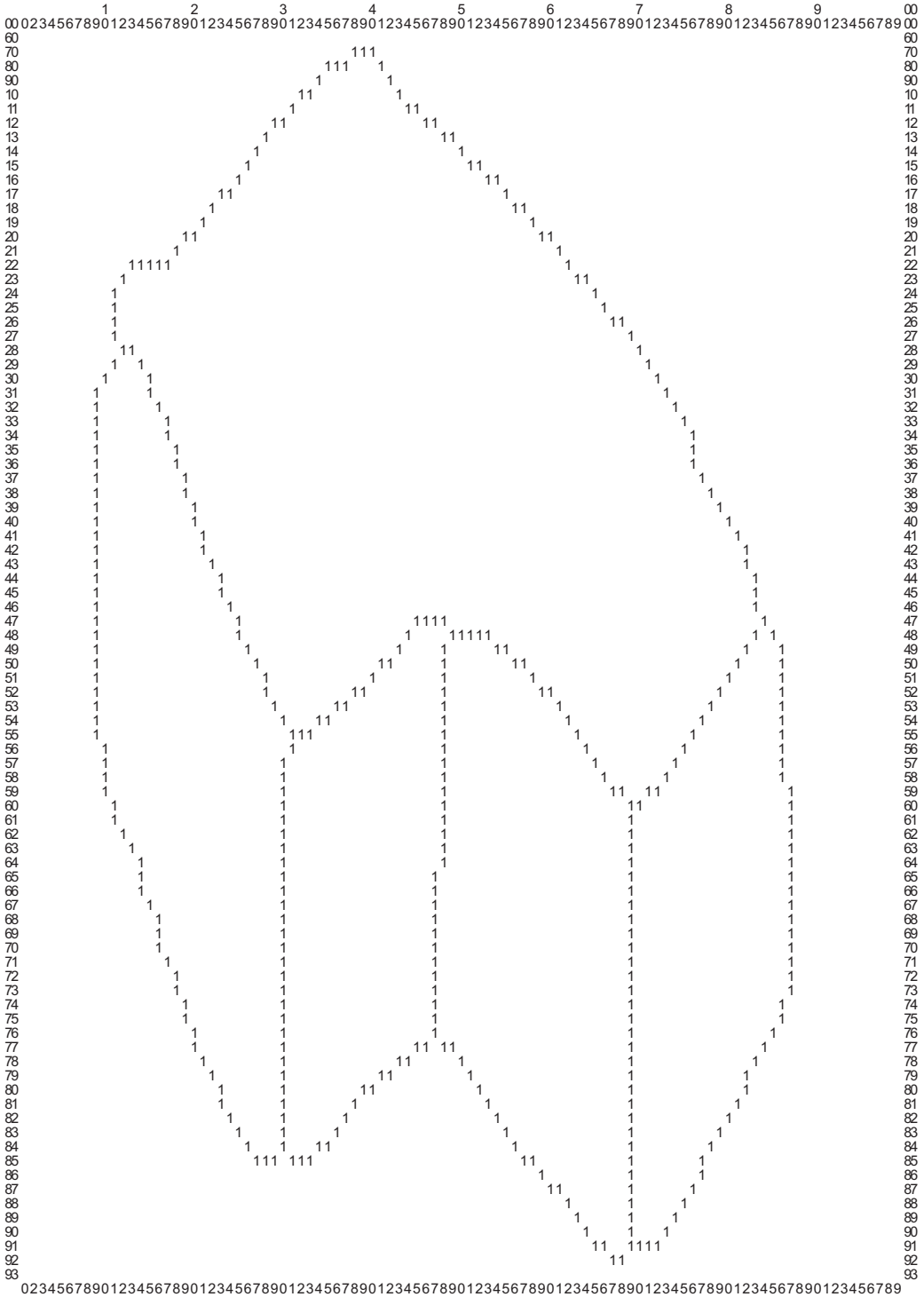

Figure 3 Raw image of Figure 2 after thinning process

Step 4: This process is repeated until the location is back to the start location i.e. at (7,38).

The first chain code series in Figure 4 shows the chain code of the boundary. The process is repeated to find the internal line.

After the traversal of boundary line is finished, the first '1' pixel found is at location (28,13) and the start code is 0. Starting from this location, the traversal process is continued until pixel '2' is found or there is no more pixels to be traversed. The second chain code series in Figure 4 shows the chain code of the first internal line starting from location (28,13) and ending at location (48,82). The process is repeated until no more '1' pixel is found. The third, fourth, and fifth series of chain code in Figure 4, which started from location (49,48), (56,31), and (61,69) respectively, represent the vertical internal line of the L-block.

The algorithm produces a series of chain code. The first one actually represents the border of the image while the rest of the series represent the internal line of the line. Figure 4 shows the complete chain code series.

Using the chain code series in Figure 4, T-junction of the image can be obtained by finding intersection of boundary chain code and internal chain code. There are two types of T-junction which are located on the boundary line and internal line. T-junction on boundary is derived first, followed by T-junction on internal line.

For every coordinate in the boundary chain code, the coordinates are compared to the first and last record of all internal line chain code. If a current boundary coordinate is adjacent to those records, then the boundary coordinate will be assigned as T-junction.

Comparing all coordinates in the current internal line chain code will derive T-junction on internal line. These values are compared to the first and last records of the

Chain Code : Boundary and Internal Line

Start From (7, 38)
00111101010110101101101110111011111111221111112122
11122222222221222222222222223233332333332333334443454
55554554555555555453434343433333434453445555655565656
56655665556566656666666666666666666666667775666770000
077077077777077077007-1

Start From (28,13)
11211212121212112112111211100707070770777000100001010110
01111111010707777777777-1

Start From (49,48) 22222222222222232222222222-1

Start From (56,31) 32222222222222222222222222-1

Start From (61,69) 222222222222222222222222222-1

Figure 4 Chain code of boundary and internal line of L-block Figure 3

rest of the internal line chain code. If adjacent coordinate is found, then the T-junction is found.

Figure 5 shows the list of T-junctions found where *PntType* is equal to 3 which indicates that the junction is located on the boundary line while 99 indicates that the point is not on the boundary.

Row	Col	PntNo	PntType
47	84	0	3
91	70	1	3
76	47	2	3
84	30	3	3
28	12	4	3
55	31	5	99
48	49	6	99
60	69	7	99

Figure 5 T-Junction table of L-block Figure 3

4.0 DERIVING REGIONS

In deriving chain code for the region, again there is a need to traverse the image from the location (0,0) and use the junction table that has just been derived. Besides, there is also a need to create a new chain code for each region. The traversal of the image is similar to the traversal of deriving T-junction except in the determination of the first test location and the direction of pixel searching. Table 3 shows the start test location. The start location is different compared to Table 2. The direction of the traversal is also in the opposite way namely in anti-clockwise direction. There are two functions of the junction table in deriving the region. The first function is to determine the number of region in the image and the other is to provide the start location of the traversal. Algorithm 2 shows the deriving region algorithm.

Table 3 Start test location (anti-clockwise direction)

Previous code	0	1	2	3	4	5	6	7
Start test location	3	4	5	6	7	0	1	2

The explanation of Algorithm 2 is based on the binary thinned image of Figure 3. The steps taken are as follows:

Step 1: In the second series of region code (Figure 6), location (91,70) is the second junction of the junction table. The next consecutive junction is at location (76,47). The chain codes between these junctions are copied into the region series.

Step 1: For every T-junction, the record of chain code between the T-junction and the next consecutive T-junction is copied into region chain code.
Step 2: Starting from the next consecutive T-junction, the connected pixel in the window 3×3 is determined using the previous location code of the current pixel as shown in Table 3.
Step 3: Starting from the start location, the window is traversed in anti-clockwise direction to find the next connected pixel. Change the pixel to '2' to indicate that the location is already traversed. Put the code of the location in the series of the region chain code.
Step 4: Move the location 0 current pixel to the next pixel found. Repeat step 2 until the traversal find pixel '2'.

Algorithm 2 Deriving region

Region chain code now contains the chain code from location (91,70) till (76,47). These chain codes are also part of the first chain code in Figure 4.

Step 2: From location (76,47), the start test location is based on the previous code (refer Table 3). Since the previous code is 5, then the start test location is at code 0.

Step 3: By searching in anti-clockwise direction, the first '1' pixel found is at location (75,47) or at code 6. The pixel is changed to '2'.

Step 4: The current location pixel is changed to the location (75,47) and the region chain code is added by the code 6. This process is repeated until the traversal back to location (91,70), i.e. the pixel 2 is found.

Figure 6 shows the series of chain code produced for region searching. There are five junctions on the boundary, namely at location (47,84), (91,70), (76,47), (84,30), and (28,12). Therefore, there are five series of region chain code derived from the traversal, as shown in Figure 6.

Based on the region chain code, location of each code will be compared to the junction list. If the location of the code is a junction, then the junction will be added in the list of junctions bounding the region. Algorithm 3 shows the algorithm to create region table.

For example, the first region is bounded from three junctions namely junction 0, 1, and 7. Junction 7 is added in the list of point in region by comparing the location of the code and the junction table. Figure 7 shows the region table corresponding to the chain code of Figure 6. Note that the order of start junction of each region is based on the sequence of the T-junction. Note also that there is one corner between location (47,84) and location (91,70). The searching of this kind of junction is beyond our discussion and can be referred in Haron [17].

The algorithm is tested on the thinned image of the L-block (Figure 3). Figure 8 shows the corresponding label of the junction and region derived. There are five T-

CHAIN CODE : REGION

Start From (47,84)
1122222222212222222222222232333323333323333344566666666666666666666666677077777777777-1

Start From (91,70)
434545555455455555555456666666666676666666666666670000101011011111110122222222222222222222222222222-1

Start From (76,47)
34343434333343445666666666666666666666666666770707070770770770001322222222222222232222222222-1

Start From (84,30)
344555565556565655665566555656665666666666666666666666677011211212121211211211211232222222222222222222222222222-1

Start From (28,12)
566677000077077707777707707700001111010101101011011101110111111112211111121221333333333333434545555555455454544454443334334343434455565556556556565656556555-1

Figure 6 Region code of L-block Figure 3

```
for every region chain code series
   for every coordinate in the chain code series
      if the coordinate match with the
         coordinate in junction table, add the
         junction in the list of junction that
         bounded the region.
      endif
   endfor
endfor
```

Algorithm 3 Region creation

List of Point in Region 0:	0	1	7		
List of Point in Region 1:	1	2	6	7	
List of Point in Region 2:	2	3	5	6	
List of Point in Region 3:	3	4	5		
List of Point in Region 4:	4	0	7	6	5

Figure 7 Region table of L-block Figure 3

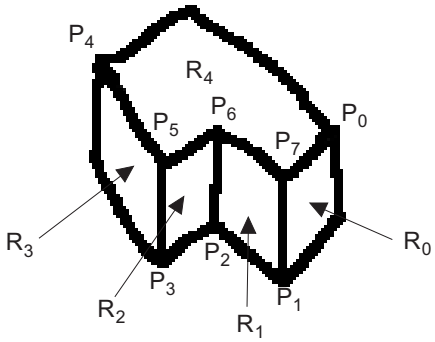

Figure 8 L-block with label

junctions extracted and they are labeled as P_0, P_1, P_2, P_3, and P_4. There are five regions extracted and they are labeled as R_0, R_1, R_2, R_3, and R_4.

5.0 DISCUSSION ON RESULTS

Results show that the list of points labeled as P_1, P_2, P_3, P_4, P_5, P_6, and P_7 shown in Figure 8 have been derived exactly as intended in the sketch. The list of points as shown in Figure 7 that bounded all three regions which are labeled as R1, R2, and R3 in Figure 8 have also been derived exactly as intended in the sketch.

The junction derivation employed in this paper can be compared to the constraint enforcement employed by Jenkins and Martin [18] of the system called Easel. The proposed constraint enforcement algorithm has been used by Grimstead [19] to derive junctions in his sketch interpretation. The L-block as shown in Figure 9 used by Grimstead [19], is used to compare the results. In terms of the number and coordinate of T-junctions, similar result has been derived. The comparison in terms of processing time taken and accuracy of the coordinate are beyond the scope of this study.

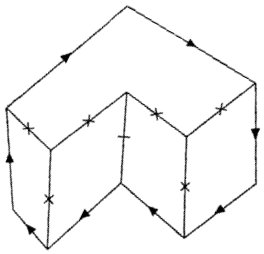

Figure 9 L-block Grimstead[19]

6.0 CONCLUSION & FUTURE WORKS

This paper shows that chain code can be used to derive geometry entities of a line drawing. The simple way to find T-junction which is by applying the principle that intersection of two lines will produce T-junction, which is applied in this work. Two algorithms have been produced to show the significance of direction of traversal in deriving chain code. The start of test location proposed in this work is capable of avoiding the traversal from missing any region or breaking the continuity of any lines. The searching of corner or V-junction between T-junction must be carried out so that the region will produce the correct shape as in the original image. The algorithm of corner detection of the chain code can be referred to Haron [17]. The chain code can also be used as input in line drawing interpretation and line labeling algorithm.

REFERENCES

[1] Freeman, H. 1974. Computer Processing of Line Drawing Images. *Computing Surveys*. 6(1): 57-97.

[2] Freeman. H. 1961. On the Encoding of Arbitrary Geometric Configurations. *IRE Trans. Electron. Comp.* EC-10: 260-268.

[3] Bribiesca, E. 2000. A Chain Code for Representing 3D Curves. *Pattern Recognition*. 33: 755-765.

[4] Bribiesca, E. 1999. A New Chain Code, Pattern Recognition. 32: 235-251.

[5] Chen, Y. W., S. C. Lee, 1991. The C-chain Code – A New Method for Coding 3D Curves. *Proceedings of Asimolar Conference on Signals, Systems & Computers*. Part 1(of 2): 472-476.

[6] Arrebola, F., A. Bandera, P. Camacho., and F. Sandoval. 1997. Corner Detection by Local Histograms of Contour Chain Code. *Electronics Letters*. 33(21): 1769-1771.

[7] O'Gorman, L. 1988. Primitives Chain Code. *International Conference on Acoustics, Speech, and Signal Processing*. ICASSP-88. 2: 792-795.

[8] Nunes, P., F. Pereira., and F. Marques. 1997. Multi-Grid Chain Coding of Binary Shapes. *Proc. Of Int. Conf. On Image Processing*. 3: 114-117, 26-29.

[9] Choo, C. Y., and H. Freeman. 1992. An Efficient Technique for Compressing Chain-coded Line Drawing Images. *Proceedings Conference on 26th Asimolar Conference on Signals, Systems and Computers*. 2: 717-720.

[10] Bribiesca, E. 2003. Scanning-curves Representation for the Coverage of Surfaces Using Chain Code. *Computer & Graphics*. 27(1): 123-132.

[11] Choo, C. Y. 1986. Line Drawing Representation Using The Polycurve Code (Chain Code). *PhD Thesis*. Rensselaer Polytechnic Institute. U.S.A.

[12] Livarinen, J. 1996. Shape Recognition of Irregular Objects. *Proceedings of SPIE-The International Society for Optical Engineering*. 2904: 25-32.

[13] Pitas, I. 1995. *Digital Image Processing Algorithm*. University Press Cambridge: Prentice Hall.

[14] Arrebola, F., A., Bandera., P. Camacho., and F. Sandoval. 1999. Corner Detection and Curve Representation by Circular Histograms of Contour Chain Code. *Electronics Letters*. 35(13): 1065-1067.

[15] Hanyu, T., S. Choi, M. Kameyama., and T. Higuchi. 1993. 3-D Object Recognition System Based on 2-D Chain Code Matching. *IEICE Trans. On Fundamentals of Electronics, Communications and Computer Sciences*. V E76-A n 6: 97-923.

[16] Maeder, A. J. 1990. Animation Techniques for Chain-coded Objects. *Proc. Of First IEEE Conf. On Visualization*. 67-73.

[17] Haron, H., D. Mohamed., and S. M. Shamsuddin. 2003. Extraction of Junctions, Lines and Regions of Irregular Line Drawing: The Chain Code Processing Algorithm. *Jurnal Teknologi*. 38 (D): 1-28.

[18] Jenkins, D. L., and R. R. Martin. 1992. Applying Constraints to Enforce Users' Intentions in Freehand 2D Sketches. *Intelligent Systems Engineering*. 1(1): 32-49.

[19] Grimstead, I. J. 1997. Interactive Sketch Input of Boundary Representation Solid Models. PhD Thesis. Univ. of Cardiff. U.K.